Introduction to Big Data Assignment-8

Name: Faizan Mulla
Roll No: 21F1003885

Problem Statement

Convert the Spark MLlib code shown at Databricks Notebook to use the CrossValidator autotuner. Report on the best performing model parameters.

Here is the link : https://docs.databricks.com/_extras/notebooks/source/decision-trees.html

---

Solution

This solution will be implemented on a **GCP Compute Engine VM** instance using the **SSH terminal** for the setup and execution of the code.

To solve the problem, we will:

1. **Set Up Google Cloud Environment**: We will configure a Compute Engine VM instance on GCP and set up the necessary environment for Spark MLlib and Python 3.12.
2. **Install Required Libraries**: We will install Spark, Python libraries (like pyspark), and necessary dependencies on the VM.
3. **Implement CrossValidator**: Modify the decision tree classifier to use **CrossValidator** and perform hyperparameter tuning.
4. **Model Evaluation**: We will evaluate the best model parameters and report them based on performance metrics like accuracy or precision.

Implementation Details

1. **Setting Up GCP Environment**
    - First, we create a **Google Compute Engine VM** instance with sufficient CPU and memory for Spark operations.
    - **SSH** into the VM to begin setting up the environment.
2. **System Setup and Python Environment**:
    - We install necessary updates and Python dependencies on the VM.
    - We create a virtual environment (*.venv*) for managing dependencies in isolation.
3. **Spark Installation**:
    - We install Apache Spark on the VM to enable distributed data processing.
    - The *CrossValidator* will be used within Spark MLlib to tune the Decision Tree parameters.
4. **Pipeline Creation and CrossValidator**:
    - The pipeline will include stages for feature transformation, the decision tree model, and cross-validation to tune the model parameters.
    - We will evaluate the model using the **MulticlassClassificationEvaluator**.
5. **Execution Process**:
    - We will run the code using **Spark** on the GCP VM instance and report the best parameters identified by CrossValidator.

Execution Process

Step 1: *VM instance configuration*

- Navigate to Compute Engine and then to VM instances.
- Main settings:
    - Name: **ibd-ga8-vm**
    - Region: **asia-south-1**
    - Zone: **asia-south-1-a**

- Keep Machine Configuration settings as default (or you can change it based on your requirement)

- Identity and Access Scopes settings (**IMP**) →
  Choose this: Allow full access to all Cloud APIs

- Firewall Settings →
  Check these boxes: Allow HTTP traffic & Allow HTTPS traffic

- Click on “Create”

Step 2: *Environment Preparation / VM setup*

Click the "SSH" button next to your VM. This opens a browser-based terminal.

Create Virtual Environment

- First install required packages:
  sudo apt-get update
  sudo apt -y upgrade
  sudo apt install -y python3.12-venv

- Create a virtual environment

python3 -m venv .venv

- Activate the virtual environment:
  source .venv/bin/activate

- Now, you have to upload 2 files → "*mnist.py*" & "*requirements.txt*"
  For this: In the top right corner, click on **"Upload Files"** and choose the Python script *'mnist.py'* and *'requirements.txt'* from your computer. These files will be in your home directory now.

Here are the contents of the requirements.file :

scikit-learn
pandas
pyspark
setuptools

Step 3: *Installing Java and Setting JAVA_HOME*

Since PySpark requires Java, you need to install Java and set the JAVA_HOME environment variable.

- Install OpenJDK 11:

  sudo apt update
  sudo apt install -y openjdk-11-jdk

- Verify Java Installation:
  java -version

- Find Java Installation Path: *readlink -f $(which java)*
- Remove */bin/java* to get JAVA_HOME
  Example: */usr/lib/jvm/java-11-openjdk-amd64*

- Set `JAVA_HOME` Environment Variable:
    - Open the shell configuration file in a text editor:
      nano ~/.bashrc
    - Add the following lines at the end of the file:
      export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
      export PATH=$JAVA_HOME/bin:$PATH
- Save and exit the editor (Ctrl + O, then Ctrl + X).
- Apply the Changes: *source ~/.bashrc*
- Verify `JAVA_HOME`: *echo $JAVA_HOME*

Step 4: *Running the Python Script*

1. Ensure Virtual Environment is activated: *source .venv/bin/activate*
2. Install Additional Dependencies (if not already installed): *pip install -r requirements.txt*
3. Execute the Python Script**:** *python3 mnist.py*

   This command runs the mnist.py script, which performs data loading, preprocessing, model training with CrossValidator, and evaluation.

4. Monitor the Execution:
   Observe the terminal for output messages indicating the progress of the script & look for messages that display the best model parameters and performance metrics

Step 5: *Cleaning Up*

1. Deactivate the Virtual Environment: *deactivate*
2. Shut Down the VM (if no longer needed):
    - In the GCP Console, navigate to Compute Engine > VM instances.
    - Click on the Stop button next to your VM instance **(ibd-ga8-vm)** to shut it down and prevent further charges.

Results

1. **Best Parameters**: After executing the code, the best model parameters selected by **CrossValidator** are displayed. These parameters could include:
    - **maxDepth**:
      Defines the maximum depth of the decision tree. It controls the complexity of the model by limiting how deep the tree can grow.
        i. A **higher value** allows the tree to learn more intricate patterns in the data but increases the risk of overfitting.
        ii. A **lower value** reduces the model's complexity and the risk of overfitting but may lead to underfitting.
    - **maxBins**:

      Specifies the maximum number of bins to discretize continuous features into for splitting at decision tree nodes.

        i. A **higher value** allows finer splits, enabling the model to capture more detailed information but requires more computation and memory.
        ii. A **lower value** reduces computational cost but may result in less granular splits, potentially impacting model accuracy.

Relevant Screenshots

Best Parameters: Screenshot of the output showing the optimal hyperparameters selected by CrossValidator: